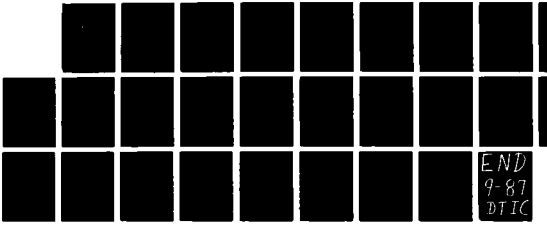
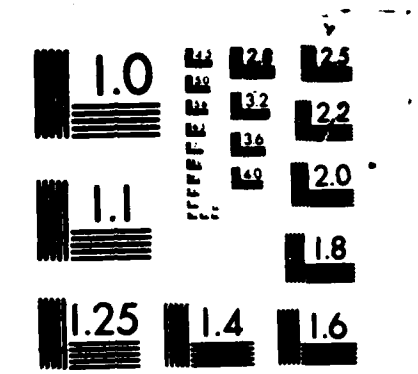
AD-A183 669 ADA (TRADEMARK) COMPILER VALIDATION SUMMARY REPORT 1/1
GOULD INC APLEX ADA CO..(U) INFORMATION SYSTEMS AND
TECHNOLOGY CENTER W-P AFB OH ADA VALI.. 25 FEB 87
UNCLASSIFIED F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DTIC FILE COPY (2)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETEING FORM
1. REPORT NUMBER | 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Ada Compiler Validation Summary Report: Gould, Inc. APLEX Ada Compiler, Version 1.0 Gould CONCEPT/32 Model 9780	5. TYPE OF REPORT & PERIOD COVERED 25 FEB 1987 to 25 FEB 1988 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Wright-Patterson	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION AND ADDRESS Ada Validation Facility ASD/SIOL Wright-Patterson AFB OH 45433-6503	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Ada Joint Program Office United States Department of Defense Washington, DC 20301-3081ASD/SIOL	12. REPORT DATE 25 FEB 1987 13. NUMBER OF PAGES 31
14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office) Wright-Patterson	15. SECURITY CLASS (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DTIC ELECTED AUG 1 2 1987 S E D

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20. If different from Report)

UNCLASSIFIED

18. SUPPLEMENTARY NOTES

19. KEYWORDS (Continue on reverse side if necessary and identify by block number)

Ada Programming language, Ada Compiler Validation Summary Report, Ada Compiler Validation Capability, ACVC, Validation Testing, Ada Validation Office, AVO, Ada Validation Facility, AVF, ANSI/MIL-STD-1815A, Ada Joint Program Office, AJPO

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Attached.

AD-A183 669

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Ada® Compiler Validation Summary Report:

Compiler Name: APLEX™ Ada Compiler, Version 1.0

Host: Gould CONCEPT/32 Model 9780 under MPX-32, Version 3.2

Target: Gould CONCEPT/32 Model 9780 under MPX-32, Version 3.2

Testing Completed 25 February 1987 Using ACVC 1.8

This report has been reviewed and is approved.

Georgeanne C Chitwood

Ada Validation Facility
Georgeanne Chitwood
ASD/SCOL
Wright-Patterson AFB OH 45433-6503

John F Kramer

Ada Validation Organization
Dr. John F. Kramer
Institute for Defense Analyses
Alexandria VA

Virginia L. Castor

Ada Joint Program Office
Virginia L. Castor
Director
Department of Defense
Washington DC

Accession For
NTIS GRA&I [X]
DTIC TAB [ ]
Unannounced [ ]
Justification
By
Distribution/
Availability Codes
Dist | Special
A-1

DTIC COPY INSPECTED

®Ada is a registered trademark of the United States Government (Ada Joint Program Office).

™APLEX is a trademark of Gould, Inc.

87 8 7 003

AVF Control Number: AVF-VSR-59.0587
86-12-18-GOU

Ada® COMPILER
VALIDATION SUMMARY REPORT:
Gould, Inc.
APLEX™ Ada Compiler, Version 1.0
Gould CONCEPT/32 Model 9780

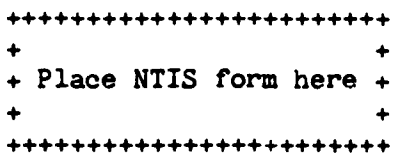
Completion of On-Site Testing:
25 February 1987

Prepared By:
Ada Validation Facility
ASD/SCOL
Wright-Patterson AFB OH 45433-6503

Prepared For:
Ada Joint Program Office
United States Department of Defense
Washington, D.C.

®Ada is a registered trademark of the United States Government (Ada Joint Program Office).

™APLEX is a trademark of Gould, Inc.


Place NTIS form here

EXECUTIVE SUMMARY

This Validation Summary Report (VSR) summarizes the results and conclusions of validation testing performed on the APLEX©Ada Compiler, Version 1.0, using Version 1.8 of the Ada® Compiler Validation Capability (ACVC). The APLEX Ada Compiler is hosted on a Gould CONCEPT/32 Model 9780 operating under MPX-32, Version 3.2. Programs processed by this compiler may be executed on a Gould CONCEPT/32 Model 9780 operating under MPX-32, Version 3.2.

On-site testing was performed 23 February 1987 through 25 February 1987 at Gould, Inc. in Ft. Lauderdale FL, under the direction of the Ada Validation Facility (AVF), according to Ada Validation Organization (AVO) policies and procedures. The AVF identified 2102 of the 2399 tests in ACVC Version 1.8 to be processed during on-site testing of the compiler. The 19 tests withdrawn at the time of validation testing, as well as the 278 executable tests that make use of floating-point precision exceeding that supported by the implementation, were not processed. After the 2102 tests were processed, results for Class A, C, D, and E tests were examined for correct execution. Compilation listings for Class B tests were analyzed for correct diagnosis of syntax and semantic errors. Compilation and link results of Class L tests were analyzed for correct detection of errors. There were 64 of the processed tests determined to be inapplicable. The remaining 2038 tests were passed.

The results of validation are summarized in the following table:

RESULT	CHAPTER												TOTAL
	2	3	4	5	6	7	8	9	10	11	12	14	
Passed	93	204	280	239	161	97	135	262	107	32	217	211	2038
Failed	0	0	0	0	0	0	0	0	0	0	0	0	0
Inapplicable	23	121	140	8	0	0	4	0	23	0	1	22	342
Withdrawn	0	5	5	0	0	1	1	2	4	0	1	0	19
TOTAL	116	330	425	247	161	98	140	264	134	32	219	233	2399

The AVF concludes that these results demonstrate acceptable conformity to ANSI/MIL-STD-1815A Ada.

®Ada is a registered trademark of the United States Government (Ada Joint Program Office).

©APLEX is a trademark of Gould, Inc.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 PURPOSE OF THIS VALIDATION SUMMARY REPORT 1-2
1.2 USE OF THIS VALIDATION SUMMARY REPORT 1-2
1.3 REFERENCES 1-3
1.4 DEFINITION OF TERMS 1-3
1.5 ACVC TEST CLASSES 1-4

CHAPTER 2 CONFIGURATION INFORMATION

2.1 CONFIGURATION TESTED 2-1
2.2 IMPLEMENTATION CHARACTERISTICS 2-2

CHAPTER 3 TEST INFORMATION

3.1 TEST RESULTS 3-1
3.2 SUMMARY OF TEST RESULTS BY CLASS 3-1
3.3 SUMMARY OF TEST RESULTS BY CHAPTER 3-2
3.4 WITHDRAWN TESTS 3-2
3.5 INAPPLICABLE TESTS 3-2
3.6 SPLIT TESTS 3-4
3.7 ADDITIONAL TESTING INFORMATION 3-4
3.7.1 Prevalidation 3-4
3.7.2 Test Method 3-5
3.7.3 Test Site 3-5

APPENDIX A DECLARATION OF CONFORMANCE

APPENDIX B APPENDIX F OF THE Ada STANDARD

APPENDIX C TEST PARAMETERS

APPENDIX D WITHDRAWN TESTS

CHAPTER 1

INTRODUCTION

This Validation Summary Report (VSR) describes the extent to which a specific Ada compiler conforms to the Ada Standard, ANSI/MIL-STD-1815A. This report explains all technical terms used within it and thoroughly reports the results of testing this compiler using the Ada Compiler Validation Capability (ACVC). An Ada compiler must be implemented according to the Ada Standard, and any implementation-dependent features must conform to the requirements of the Ada Standard. The Ada Standard must be implemented in its entirety, and nothing can be implemented that is not in the Standard.

Even though all validated Ada compilers conform to the Ada Standard, it must be understood that some differences do exist between implementations. The Ada Standard permits some implementation dependencies--for example, the maximum length of identifiers or the maximum values of integer types. Other differences between compilers result from characteristics of particular operating systems, hardware, or implementation strategies. All of the dependencies observed during the process of testing this compiler are given in this report.

The information in this report is derived from the test results produced during validation testing. The validation process includes submitting a suite of standardized tests, the ACVC, as inputs to an Ada compiler and evaluating the results. The purpose of validating is to ensure conformity of the compiler to the Ada Standard by testing that the compiler properly implements legal language constructs and that it identifies and rejects illegal language constructs. The testing also identifies behavior that is implementation dependent but permitted by the Ada Standard. Six classes of tests are used. These tests are designed to perform checks at compile time, at link time, and during execution.

INTRODUCTION

1.1 PURPOSE OF THIS VALIDATION SUMMARY REPORT

This VSR documents the results of the validation testing performed on an Ada compiler. Testing was carried out for the following purposes:

- To attempt to identify any language constructs supported by the compiler that do not conform to the Ada Standard
- To attempt to identify any unsupported language constructs required by the Ada Standard
- To determine that the implementation-dependent behavior is allowed by the Ada Standard

Testing of this compiler was conducted by SofTech, Inc., under the direction of the AVF according to policies and procedures established by the Ada Validation Organization (AVO). On-site testing was conducted from 23 February 1987 through 25 February 1987 at Gould, Inc. in Ft. Lauderdale FL.

1.2 USE OF THIS VALIDATION SUMMARY REPORT

Consistent with the national laws of the originating country, the AVO may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of this validation apply only to the computers, operating systems, and compiler versions identified in this report.

The organizations represented on the signature page of this report do not represent or warrant that all statements set forth in this report are accurate and complete, or that the subject compiler has no nonconformities to the Ada Standard other than those presented. Copies of this report are available to the public from:

Ada Information Clearinghouse
Ada Joint Program Office
OUSDRE
The Pentagon, Rm 3D-139 (Fern Street)
Washington DC 20301-3081

or from:

Ada Validation Facility
ASD/SCOL
Wright-Patterson AFB OH 45433-6503

Questions regarding this report or the validation test results should be directed to the AVF listed above or to:

Ada Validation Organization
Institute for Defense Analyses
1801 North Beauregard Street
Alexandria VA 22311

1.3 REFERENCES

1. Reference Manual for the Ada Programming Language, ANSI/MIL-STD-1815A, FEB 1983.

2. Ada Validation Organization: Procedures and Guidelines, Ada Joint Program Office, 1 JAN 1987.

3. Ada Compiler Validation Capability Implementers' Guide, SofTech, Inc., DEC 1984.

1.4 DEFINITION OF TERMS

ACVC	The Ada Compiler Validation Capability. A set of programs that evaluates the conformity of a compiler to the Ada language specification, ANSI/MIL-STD-1815A.
Ada Standard	ANSI/MIL-STD-1815A, February 1983.
Applicant	The agency requesting validation.
AVF	The Ada Validation Facility. In the context of this report, the AVF is responsible for conducting compiler validations according to established policies and procedures.
AVO	The Ada Validation Organization. In the context of this report, the AVO is responsible for setting procedures for compiler validations.
Compiler	A processor for the Ada language. In the context of this report, a compiler is any language processor, including cross-compilers, translators, and interpreters.
Failed test	A test for which the compiler generates a result that demonstrates nonconformity to the Ada Standard.
Host	The computer on which the compiler resides.

Inapplicable test	A test that uses features of the language that a compiler is not required to support or may legitimately support in a way other than the one expected by the test.
Passed test	A test for which a compiler generates the expected result.
Target	The computer for which a compiler generates code.
Test	A program that checks a compiler's conformity regarding a particular feature or features to the Ada Standard. In the context of this report, the term is used to designate a single test, which may comprise one or more files.
Withdrawn test	A test found to be incorrect and not used to check conformity to the Ada language specification. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains illegal or erroneous use of the language.

1.5 ACVC TEST CLASSES

Conformity to the Ada Standard is measured using the ACVC. The ACVC contains both legal and illegal Ada programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable, and special program units are used to report their results during execution. Class B tests are expected to produce compilation errors. Class L tests are expected to produce link errors.

Class A tests check that legal Ada programs can be successfully compiled and executed. However, no checks are performed during execution to see if the test objective has been met. For example, a Class A test checks that reserved words of another language (other than those already reserved in the Ada language) are not treated as reserved words by an Ada compiler. A Class A test is passed if no errors are detected at compile time and the program executes to produce a PASSED message.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that every syntax or semantic error in the test is detected. A Class B test is passed if every illegal construct that it contains is detected by the compiler.

Class C tests check that legal Ada programs can be correctly compiled and executed. Each Class C test is self-checking and produces a PASSED, FAILED, or NOT APPLICABLE message indicating the result when it is executed.

Class D tests check the compilation and execution capacities of a compiler. Since there are no capacity requirements placed on a compiler by the Ada Standard for some parameters--for example, the number of identifiers

permitted in a compilation or the number of units in a library--a compiler may refuse to compile a Class D test and still be a conforming compiler. Therefore, if a Class D test fails to compile because the capacity of the compiler is exceeded, the test is classified as inapplicable. If a Class D test compiles successfully, it is self-checking and produces a PASSED or FAILED message during execution.

Each Class E test is self-checking and produces a NOT APPLICABLE, PASSED, or FAILED message when it is compiled and executed. However, the Ada Standard permits an implementation to reject programs containing some features addressed by Class E tests during compilation. Therefore, a Class E test is passed by a compiler if it is compiled successfully and executes to produce a PASSED message, or if it is rejected by the compiler for an allowable reason.

Class L tests check that incomplete or illegal Ada programs involving multiple, separately compiled units are detected and not allowed to execute. Class L tests are compiled separately and execution is attempted. A Class L test passes if it is rejected at link time--that is, an attempt to execute the main program must generate an error message before any declarations in the main program or any units referenced by the main program are elaborated.

Two library units, the package REPORT and the procedure CHECK_FILE, support the self-checking features of the executable tests. The package REPORT provides the mechanism by which executable tests report PASSED, FAILED, or NOT APPLICABLE results. It also provides a set of identity functions used to defeat some compiler optimizations allowed by the Ada Standard that would circumvent a test objective. The procedure CHECK_FILE is used to check the contents of text files written by some of the Class C tests for chapter 14 of the Ada Standard. The operation of these units is checked by a set of executable tests. These tests produce messages that are examined to verify that the units are operating correctly. If these units are not operating correctly, then the validation is not attempted.

The text of the tests in the ACVC follow conventions that are intended to ensure that the tests are reasonably portable without modification. For example, the tests make use of only the basic set of 55 characters, contain lines with a maximum length of 72 characters, use small numeric values, and place features that may not be supported by all implementations in separate tests. However, some tests contain values that require the test to be customized according to implementation-specific values--for example, an illegal file name. A list of the values used for this validation is provided in Appendix C.

A compiler must correctly process each of the tests in the suite and demonstrate conformity to the Ada Standard by either meeting the pass criteria given for the test or by showing that the test is inapplicable to the implementation. Any test that was determined to contain an illegal language construct or an erroneous language construct is withdrawn from the ACVC and, therefore, is not used in testing a compiler. The tests withdrawn at the time of validation are given in Appendix D.

CHAPTER 2

CONFIGURATION INFORMATION

2.1 CONFIGURATION TESTED

The candidate compilation system for this validation was tested under the following configuration:

Compiler: APLEX Ada Compiler, Version 1.0

ACVC Version: 1.8

Certificate Expiration Date: 2 April 1988

Host Computer:

Machine:	Gould CONCEPT/32 Model 9780
Operating System:	MPX-32, Version 3.2
Memory Size:	16 megabytes

Target Computer:

Machine:	Gould CONCEPT/32 Model 9780
Operating System:	MPX-32, Version 3.2
Memory Size:	16 megabytes

2.2 IMPLEMENTATION CHARACTERISTICS

One of the purposes of validating compilers is to determine the behavior of a compiler in those areas of the Ada Standard that permit implementations to differ. Class D and E tests specifically check for such implementation differences. However, tests in other classes also characterize an implementation. This compiler is characterized by the following interpretations of the Ada Standard:

- Capacities.

  The compiler correctly processes tests containing loop statements nested to 65 levels, block statements nested to 65 levels, and recursive procedures separately compiled as subunits nested to 17 levels. It correctly processes a compilation containing 723 variables in the same declarative part. (See tests D55A03A..H (8 tests), D56001B, D64005E..G (3 tests), and D29002K.)

- Universal integer calculations.

  An implementation is allowed to reject universal integer calculations having values that exceed SYSTEM.MAX_INT. This implementation does not reject such calculations and processes them correctly. (See tests D4A002A, D4A002B, D4A004A, and D4A004B.)

- Predefined types.

  This implementation does not support additional predefined types in the package STANDARD. (See tests B86001C and B86001D.)

- Based literals.

  An implementation is allowed to reject a based literal with a value exceeding SYSTEM.MAX_INT during compilation, or it may raise NUMERIC_ERROR or CONSTRAINT_ERROR during execution. This implementation raises NUMERIC_ERROR during execution. (See test E24101A.)

- Array types.

  An implementation is allowed to raise NUMERIC_ERROR or CONSTRAINT_ERROR for an array having a 'LENGTH that exceeds STANDARD.INTEGER'LAST and/or SYSTEM.MAX_INT.

  A packed BOOLEAN array having a 'LENGTH exceeding INTEGER'LAST raises NUMERIC_ERROR when the array type is declared. (See test C52103X.)

A packed two-dimensional BOOLEAN array with more than INTEGER'LAST components raises NUMERIC_ERROR when the array type is declared. (See test C52104Y.)

A null array with one dimension of length greater than INTEGER'LAST may raise NUMERIC_ERROR or CONSTRAINT_ERROR either when declared or assigned. Alternatively, an implementation may accept the declaration. However, lengths must match in array slice assignments. This implementation raises NUMERIC_ERROR when the array type is declared. (See test E52103Y.)

In assigning one-dimensional array types, the expression appears to be evaluated in its entirety before CONSTRAINT_ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. In assigning two-dimensional array types, the expression does not appear to be evaluated in its entirety before CONSTRAINT_ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)

- Discriminated types.

  During compilation, an implementation is allowed to either accept or reject an incomplete type with discriminants that is used in an access type definition with a compatible discriminant constraint. This implementation accepts such subtype indications. (See test E38104A.)

  In assigning record types with discriminants, the expression appears to be evaluated in its entirety before CONSTRAINT_ERROR is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)

- Aggregates.

  In the evaluation of a multi-dimensional aggregate, the order in which choices are evaluated and index subtype checks are made appears to depend upon the aggregate itself. (See tests C43207A and C43207B.)

  In the evaluation of an aggregate containing subaggregates, all choices are not evaluated before being checked for identical bounds. (See test E43211B.)

  All choices are evaluated before CONSTRAINT_ERROR is raised if a bound in a nonnull range of a nonnull aggregate does not belong to an index subtype. (See test E43211B.)

. Functions.

An implementation may allow the declaration of a parameterless function and an enumeration literal having the same profile in the same immediate scope, or it may reject the function declaration. If it accepts the function declaration, the use of the enumeration literal's identifier denotes the function. This implementation rejects the declaration. (See test E66001D.)

. Representation clauses.

The Ada Standard does not require an implementation to support representation clauses. If a representation clause is not supported, then the implementation must reject it. While the operation of representation clauses is not checked by Version 1.8 of the ACVC, they are used in testing other language features. This implementation accepts 'SIZE and 'STORAGE_SIZE for tasks; it rejects 'STORAGE_SIZE for collections and 'SMALL clauses. Enumeration representation clauses, including those that specify noncontiguous values, appear not to be supported. (See tests C55B16A, C87B62A, C87B62B, C87B62C, and BC1002A.)

. Pragmas.

The pragma INLINE is not supported for procedures or functions. (See tests CA3004E and CA3004F.)

. Input/output.

The package SEQUENTIAL_IO cannot be instantiated with unconstrained array types and record types with discriminants. The package DIRECT_IO cannot be instantiated with unconstrained array types and record types with discriminants without defaults. (See tests AE2101C, AE2101H, CE2201D, CE2201E, and CE2401D.)

An existing text file can be opened in OUT_FILE mode and can be created in both OUT_FILE and IN_FILE modes. (See test EE3102C.)

Only one internal file can be associated with each external file for text I/O for both reading and writing. (See tests CE3111A..E (5 tests).)

Only one internal file can be associated with each external file for sequential I/O for both reading and writing. (See tests CE2107A..F (6 tests).)

Only one internal file can be associated with each external file for direct I/O for both reading and writing. (See tests CE2107A..F (6 tests).)

Temporary sequential files are given a name. Temporary direct files are given a name. Temporary files given names are not deleted when they are closed. (See tests CE2108A and CE2108C.)

- Generics.

  Generic subprogram declarations and bodies cannot be compiled in separate compilations. (See test CA2009F.)

  Generic package declarations and bodies cannot be compiled in separate compilations. (See tests CA2009C and BC3205D.)

CHAPTER 3

TEST INFORMATION

3.1 TEST RESULTS

Version 1.8 of the ACVC contains 2399 tests. When validation testing of APLEX Ada Compiler was performed, 19 tests had been withdrawn. The remaining 2380 tests were potentially applicable to this validation. The AVF determined that 342 tests were inapplicable to this implementation, and that the 2038 applicable tests were passed by the implementation.

The AVF concludes that the testing results demonstrate acceptable conformity to the Ada Standard.

3.2 SUMMARY OF TEST RESULTS BY CLASS

RESULT	TEST CLASS						TOTAL
	A	B	C	D	E	L	
Passed	67	860	1052	17	11	31	2038
Failed	0	0	0	0	0	0	0
Inapplicable	2	7	316	0	2	15	342
Withdrawn	0	7	12	0	0	0	19
TOTAL	69	874	1380	17	13	46	2399

3.3 SUMMARY OF TEST RESULTS BY CHAPTER

RESULT	CHAPTER												TOTAL
	2	3	4	5	6	7	8	9	10	11	12	14	
Passed	93	204	280	239	161	97	135	262	107	32	217	211	2038
Failed	0	0	0	0	0	0	0	0	0	0	0	0	0
Inapplicable	23	121	140	8	0	0	4	0	23	0	1	22	342
Withdrawn	0	5	5	0	0	1	1	2	4	0	1	0	19
TOTAL	116	330	425	247	161	98	140	264	134	32	219	233	2399

3.4 WITHDRAWN TESTS

The following 19 tests were withdrawn from ACVC Version 1.8 at the time of this validation:

C32114A	C41404A	B74101B	BC3204C
B33203C	B45116A	C87B50A	
C34018A	C48008A	C92005A	
C35904A	B49006A	C940ACA	
B37401A	B4A010C	CA3005A..D (4 tests)	

See Appendix D for the reason that each of these tests was withdrawn.

3.5 INAPPLICABLE TESTS

Some tests do not apply to all compilers because they make use of features that a compiler is not required by the Ada Standard to support. Others may depend on the result of another test that is either inapplicable or withdrawn. For this validation attempt, 342 tests were inapplicable for the reasons indicated:

- C34001D, B52004E, B55B09D, and C55B07B use SHORT_INTEGER which is not supported by this compiler.
- C34001E, B52004D, B55B09C, and C55B07A use LONG_INTEGER which is not supported by this compiler.
- C34001F and C35702A use SHORT_FLOAT which is not supported by this compiler.

- C34001G and C35702B use LONG_FLOAT which is not supported by this compiler.

- C52008B declares a record type with four discriminants of type integer and having default values. The type may be used in the declaration of unconstrained objects, but the size of these objects exceeds the maximum object size of this implementation, and NUMERIC_ERROR is raised.

- C55B16A makes use of an enumeration representation clause containing noncontiguous values which is not supported by this compiler.

- B86001D requires a predefined numeric type other than those defined by the Ada language in package STANDARD. There is no such type for this implementation.

- C86001F redefines package SYSTEM, but TEXT_IO is made obsolete by this new definition in this implementation and the test cannot be executed since the package REPORT is dependent on the package TEXT_IO.

- C87B62B..C (2 tests) use length clauses which are not supported by this compiler. The length clauses 'STORAGE_SIZE for access types and 'SMALL are rejected during compilation.

- BA1011C, CA1012A, CA2009C, CA2009F, LA5008A..H (8 tests), LA5008J, LA5008M, LA5008N, and BC3205D compile generic specifications and bodies in separate compilations which is not supported by this compiler.

- CA3004E, EA3004C, and LA3004A use INLINE pragma for procedures which is not supported by this compiler.

- CA3004F, EA3004D, and LA3004B use INLINE pragma for functions which is not supported by this compiler.

- LA5008I and LA5008K are inapplicable because, in this implementation, a generic unit is made obsolete by the recompilation of a unit on which the generic body (but not the specification) depends. Since this implementation does not support separate compilation of generic unit specifications and bodies, a generic specification must be considered obsolete whenever the body is found to be obsolete. These tests should report at link time that the body of a generic unit is obsolete. However, a compile-time error message reports that the generic unit is obsolete.

- AE2101C, CE2201D, and CE2201E use an instantiation of package SEQUENTIAL_IO with unconstrained array types which is not supported by this compiler.

- AE2101H and CE2401D use an instantiation of package DIRECT_IO with unconstrained array types which is not supported by this compiler.

- CE2107A..F (6 tests), CE2110B, CE2111D, CE2111H, CE3111A..E (5 tests), CE3114B, and CE3115A are inapplicable because multiple internal files cannot be associated with the same external file. The proper exception is raised when multiple access is attempted.

- CE3605A attempts to output a line 360 characters long. This implementation limits output lines to 253 characters. Placing a limitation on the length of a text line is an acceptable implementation restriction. An attempt to write the 254th character results in the exception USE_ERROR being raised.

- The following 278 tests require a floating-point accuracy that exceeds the maximum of 6 supported by the implementation:

C24113C..Y (23 tests)	C35708C..Y (23 tests)	C45421C..Y (23 tests)
C35705C..Y (23 tests)	C35802C..Y (23 tests)	C45424C..Y (23 tests)
C35706C..Y (23 tests)	C45241C..Y (23 tests)	C45521C..Z (24 tests)
C35707C..Y (23 tests)	C45321C..Y (23 tests)	C45621C..Z (24 tests)

3.6 SPLIT TESTS

If one or more errors do not appear to have been detected in a Class B test because of compiler error recovery, then the test is split into a set of smaller tests that contain the undetected errors. These splits are then compiled and examined. The splitting process continues until all errors are detected by the compiler or until there is exactly one error per split. Any Class A, Class C, or Class E test that cannot be compiled and executed because of its size is split into a set of smaller subtests that can be processed.

Splits were required for six Class B tests:

BA3006A	BA3007B	BA3008B
BA3006B	BA3008A	BA3013A

3.7 ADDITIONAL TESTING INFORMATION

3.7.1 Prevalidation

Prior to validation, a set of test results for ACVC Version 1.8 produced by the APLEX Ada Compiler was submitted to the AVF by the applicant for review. Analysis of these results demonstrated that the compiler successfully passed all applicable tests, and that the compiler exhibited the expected behavior on all inapplicable tests.

3.7.2 Test Method

Testing of the APLEX Ada Compiler using ACVC Version 1.8 was conducted on-site by a validation team from the AVF. The configuration consisted of a Gould CONCEPT/32 Model 9780 operating under MPX-32, Version 3.2.

A magnetic tape containing all tests except for withdrawn tests and tests requiring unsupported floating-point precisions was taken on-site by the validation team for processing. Tests that make use of implementation-specific values were customized before being written to the magnetic tape. Tests requiring splits during the prevalidation testing were included in their split form on the magnetic tape.

The contents of the magnetic tape were loaded directly onto the host computer. After the test files were loaded to disk, the full set of tests was compiled, linked, and executed as appropriate on a Gould CONCEPT/32 Model 9780. Results were printed from the Gould CONCEPT/32 Model 9780.

The compiler was tested using command scripts provided by Gould, Inc. and reviewed by the validation team. The following options were in effect for testing:

Option	Effect
listing	Generates compilation listing. Default is to have listing disabled.
list_name	Allows specification of the compilation listing filename. Default is to have list_name disabled.

Test output, compilation listings, and job logs were captured on magnetic tape and archived at the AVF. The listings examined on-site by the validation team were also archived.

3.7.3 Test Site

The validation team arrived at Gould, Inc. in Ft. Lauderdale FL on 22 February 1987, and departed after testing was completed on 25 February 1987.

APPENDIX A

DECLARATION OF CONFORMANCE

Gould, Inc. has submitted the following declaration of conformance concerning the APLEX Ada Compiler.

DECLARATION OF CONFORMANCE

Compiler Implementor: TeleSoft, Inc.
Ada® Validation Facility: ASD/SCOL, Wright-Patterson AFB, OH
Ada Compiler Validation Capability (ACVC) Version: 1.8

Base Configuration

Base Compiler Name: APLEX™ Ada Compiler Version: 1.0
Host Architecture ISA: Gould CONCEPT/32 Model 9780 OS&VER #: MPX, Version 3.2
Target Architecture ISA: Gould CONCEPT/32 Model 9780 OS&VER #: MPX, Version 3.2

Implementor's Declaration

I, the undersigned, representing TeleSoft, Inc., have implemented no deliberate extensions to the Ada Language Standard ANSI/MIL-STD-1815A in the compiler listed in this declaration. I declare that Gould Inc. is the owner of record of the Ada language compiler listed above and, as such, is responsible for maintaining said compiler in conformance to ANSI/MIL-STD-1815A. All certificates and registrations for Ada language compiler listed in this declaration shall be made only in the owner's corporate name.

Date: 3/6/87

TeleSoft, Inc.
Ray Parra, Director of Contracts/Legal

Owner's Declaration

I, the undersigned, representing Gould Inc., take full responsibility for implementation and maintenance of the Ada compiler listed above, and agree to the public disclosure of the final Validation Summary Report. I further agree to continue to comply with the Ada trademark policy, as defined by the Ada Joint Program Office. I declare that all of the Ada language compilers listed, and their host/target performance are in compliance with the Ada Language Standard ANSI/MIL-STD-1815A. I have reviewed the Validation Summary Report for the compiler and concur with the contents.

Mary F. Macomber Date: 3/5/87

Gould Inc.
Mary F. Macomber, Manager, Major Corporate Agreements

®Ada is a registered trademark of the United States Government (Ada Joint Program Office)

™APLEX is a trademark of Gould Inc.

APPENDIX B

APPENDIX F OF THE Ada STANDARD

The only allowed implementation dependencies correspond to implementation-dependent pragmas, to certain machine-dependent conventions as mentioned in chapter 13 of MIL-STD-1815A, and to certain allowed restrictions on representation clauses. The implementation-dependent characteristics of the APLEX Ada Compiler, Version 1.0, are described in the following sections which discuss topics in Appendix F of the Ada Language Reference Manual (ANSI/MIL-STD-1815A). Implementation-specific portions of the package STANDARD are also included in this appendix.

```
package STANDARD is

    ...

    type INTEGER is range -2_147_483_648 .. 2_147_483_647;

    type FLOAT is digits 6 range -7.23698E+75 .. 7.23698E+75;

    type DURATION is delta 2#1.0#E-14 range -86_400.0 .. 86_400.0;

    ...

end STANDARD;
```

1. Implementation Dependent Pragmas

There is one implementation-defined pragma, COMMENT. It has the form:

```
pragma COMMENT(<string_literal>);
```

It may only appear within a compilation unit and has the effect of embedding the given sequence of characters in the object code of the compilation unit.

2. Implementation Dependent Attributes

There are no implementation dependent attributes.

3. Specification of Package SYSTEM

```
--Pragma Comment("This is an unpublished work written by TeleSoft");
--Pragma Comment("Copyright 1984, 1985 TeleSoft. All rights reserved");
--
--   Change History.
--
-- ps 2.02.85 Original version
-- ps 2.09.85 Additions for tasking support
-- ps 3.05.85 Modification for 32 bit integer support
-- ps 3.18.85 Added system.subprogram_value
-- ps 3.26.85 Modified Priority to be range 1..1 to fix PAR 1684
-- ps 5.17.85 Modified definitions of delta and fine_delta to use
--            be type float instead of integer.
-- ps 6.19.85 Amended definition of "fine_delta" to be an exact binary
--            number.
-- ps 6.27.85 Amended the definition of "priority" to be non-null
--

With Gould_names;
package System is

  Pragma Elaborate(Gould_names);

  type Address is private;

  type Name     is (Gould_UTX, Gould_MPX);
  System_Name   : constant name := Name'Val(Gould_names.sys_name);

  Storage_Unit : constant := 8;
  Memory_Size  : constant := 2**24-1;
```

```
   -- System-Dependent Declarations

   subtype byte       is integer range      0 .. 2** 8-1;
   subtype integer_16 is integer range -2**15 .. 2**15-1;
   subtype integer_32 is integer; -- range -2**31 .. 2**31-1;
   --subtype integer_64 is integer_32;

   -- System-Dependent Named Number

   Min_Int      : constant := -2**31;
   Max_Int      : constant := +2**31-1;
   Max_Digits   : constant := 6;
   Max_Mantissa : constant := 30;
   Fine_delta   : constant := 1.0 / (2.0 ** (Max_Mantissa - 1));
   Tick         : constant := 1.0 / (2.0 ** 14);


   -- Other System-Dependent Declarations
   Max_Object_Size    : CONSTANT  := Max_Int;
   Max_Record_Count   : CONSTANT  := Max_Int;
   Max_Text_Io_Count  : CONSTANT  := Max_Int-1;
   Max_Text_Io_Field  : CONSTANT  := 1000;

   subtype Priority is integer range 0 .. 255;


   Null_address : constant address;

   type reg_array is array (0..7) of integer_32;

   type subprogram_value is record
                              base_regs: reg_array;
                           end record;


private

   type Address is new integer_32;
   Null_address : constant address := 0;

end System;

package body system is

begin
   null;
end;
```

```
Package Gould_names is

   Function Sys_name return integer;

end Gould_names;


Package Body Gould_names is

   Function Sys_name return integer is
   begin
     Return 0;
   end;

end Gould_names;
```

4. Restrictions on Representation Clauses

   The Compiler supports the following representation clauses:

   Length Clauses: for tasks 'STORAGE_SIZE (LRM 13.2(c))
   Length Clauses: for the attribute 'SIZE (LRM 13.2(a))
   Address Clauses: for objects and entries (LRM 13.5)

5. Implementation dependent naming conventions

   There are no implementation-generated names denoting implementation dependent components.

6. Expressions that appear in address specifications are interpreted as the first storage unit of the object.

7. Restrictions on Unchecked Conversions

   Unchecked conversions are allowed between variables of types (or subtypes) T1 and T2 provided that 1) they have the same static size, 2) they are not unconstrained array types, and 3) they are not private (unless they are subtypes of or are derived from a private type SYSTEM.ADDRESS).

8. I/O Package Characteristics

   Instantiations of DIRECT_IO and SEQUENTIAL_IO are supported with the following exceptions:

   * Unconstrained array types.
   * Unconstrainted types with discriminants without default values.
   * Multiple internal files may not be associated with the external file for reading and writing.

* In DIRECT_IO the type COUNT is defined as follow:

```
type COUNT is range 0..2_147_483_647;
```

* In TEXT_IO the type COUNT is defined as follows:

```
type COUNT is range 0..2_147_483_645;
```

* In TEXT_IO the subtype FIELD is defined as follows:

```
subtype FIELD is INTEGER range 0..1000;
```

* The line length limit for the target is 253 characters.

APPENDIX C

TEST PARAMETERS

Certain tests in the ACVC make use of implementation-dependent values, such as the maximum length of an input line and invalid file names. A test that makes use of such values is identified by the extension .TST in its file name. Actual values to be substituted are represented by names that begin with a dollar sign. A value must be substituted for each of these names before the test is run. The values used for this validation are given below.

Name and Meaning	Value
$BIG_ID1 Identifier the size of the maximum input line length with varying last character.	(1..199 => 'A', 200 => '1')
$BIG_ID2 Identifier the size of the maximum input line length with varying last character.	(1..199 => 'A', 200 => '2')
$BIG_ID3 Identifier the size of the maximum input line length with varying middle character.	(1..100 | 102..200 => 'A', 101 => '3')
$BIG_ID4 Identifier the size of the maximum input line length with varying middle character.	(1..100 | 102..200 => 'A', 101 => '4')
$BIG_INT_LIT An integer literal of value 298 with enough leading zeroes so that it is the size of the maximum line length.	(1..197 => '0', 198..200 => "298")

TEST PARAMETERS

Name and Meaning	Value
$BIG_REAL_LIT A real literal that can be either of floating- or fixed-point type, has value 690.0, and has enough leading zeroes to be the size of the maximum line length.	(1..194 => '0', 195..200 => "69.0E1")
$BLANKS A sequence of blanks twenty characters fewer than the size of the maximum line length.	(1..180 => ' ')
$COUNT_LAST A universal integer literal whose value is TEXT_IO.COUNT'LAST.	2_147_483_645
$EXTENDED_ASCII_CHARS A string literal containing all the ASCII characters with printable graphics that are not in the basic 55 Ada character set.	"abcdefghijklmnopqrstuvwxyz!" & "$%?@[\]^`{}~"
$FIELD_LAST A universal integer literal whose value is TEXT_IO.FIELD'LAST.	1000
$FILE_NAME_WITH_BAD_CHARS An illegal external file name that either contains invalid characters, or is too long if no invalid characters exist.	"X}]!/@%#$^&~Y"
$FILE_NAME_WITH_WILD_CARD_CHAR An external file name that either contains a wild card character, or is too long if no wild card character exists.	"XYZ*"
$GREATER_THAN_DURATION A universal real value that lies between DURATION'BASE'LAST and DURATION'LAST if any, otherwise any value in the range of DURATION.	100_000.0
$GREATER_THAN_DURATION_BASE_LAST The universal real value that is greater than DURATION'BASE'LAST, if such a value exists.	10_000_000.0

Name and Meaning	Value
$ILLEGAL_EXTERNAL_FILE_NAME1 An illegal external file name.	"BAD_CHARACTER%/*^"
$ILLEGAL_EXTERNAL_FILE_NAME2 An illegal external file name that is different from $ILLEGAL_EXTERNAL_FILE_NAME1.	(1..17 => 'A')
$INTEGER_FIRST The universal integer literal expression whose value is INTEGER'FIRST.	-2_147_483_648
$INTEGER_LAST The universal integer literal expression whose value is INTEGER'LAST.	2_147_483_647
$LESS_THAN_DURATION A universal real value that lies between DURATION'BASE'FIRST and DURATION'FIRST if any, otherwise any value in the range of DURATION.	-100_000.0
$LESS_THAN_DURATION_BASE_FIRST The universal real value that is less than DURATION'BASE'FIRST, if such a value exists.	-10_000_000.0
$MAX_DIGITS The universal integer literal whose value is the maximum digits supported for floating-point types.	6
$MAX_IN_LEN The universal integer literal whose value is the maximum input line length permitted by the implementation.	200
$MAX_INT The universal integer literal whose value is SYSTEM.MAX_INT.	2_147_483_647

TEST PARAMETERS

Name and Meaning	Value
$NAME A name of a predefined numeric type other than FLOAT, INTEGER, SHORT_FLOAT, SHORT_INTEGER, LONG_FLOAT, or LONG_INTEGER if one exists, otherwise any undefined name.	LONG_LONG_INTEGER
$NEG_BASED_INT A based integer literal whose highest order nonzero bit falls in the sign bit position of the representation for SYSTEM.MAX_INT.	16#FFFFFFFE#
$NON_ASCII_CHAR_TYPE An enumerated type definition for a character type whose literals are the identifier NON_NULL and all non-ASCII characters with printable graphics.	(NON_NULL)

APPENDIX D

WITHDRAWN TESTS

Some tests are withdrawn from the ACVC because they do not conform to the Ada Standard. The following 19 tests had been withdrawn at the time of validation testing for the reasons indicated. A reference of the form "AI-ddddd" is to an Ada Commentary.

- C32114A: An unterminated string literal occurs at line 62.
- B33203C: The reserved word "IS" is misspelled at line 45.
- C34018A: The call of function G at line 114 is ambiguous in the presence of implicit conversions.
- C35904A: The elaboration of subtype declarations SFX3 and SFX4 may raise NUMERIC_ERROR instead of CONSTRAINT_ERROR as expected in the test.
- B37401A: The object declarations at lines 126 through 135 follow subprogram bodies declared in the same declarative part.
- C41404A: The values of 'LAST and 'LENGTH are incorrect in the if statements from line 74 to the end of the test.
- B45116A: ARRPRIBL1 and ARRPRIBL2 are initialized with a value of the wrong type--PRIBOOL_TYPE instead of ARRPRIBOOL_TYPE--at line 41.
- C48008A: The assumption that evaluation of default initial values occurs when an exception is raised by an allocator is incorrect according to AI-00397.
- B49006A: Object declarations at lines 41 and 50 are terminated incorrectly with colons, and end case; is missing from line 42.
- B4A010C: The object declaration in line 18 follows a subprogram body of the same declarative part.

WITHDRAWN TESTS

- B74101B: The begin at line 9 causes a declarative part to be treated as a sequence of statements.

- C87B50A: The call of "/=" at line 31 requires a use clause for package A.

- C92005A: The "/=" for type PACK.BIG_INT at line 40 is not visible without a use clause for the package PACK.

- C940ACA: The assumption that allocated task TT1 will run prior to the main program, and thus assign SPYNUMB the value checked for by the main program, is erroneous.

- CA3005A..D (4 tests): No valid elaboration order exists for these tests.

- BC3204C: The body of BC3204C0 is missing.

END

9-87

DTIC